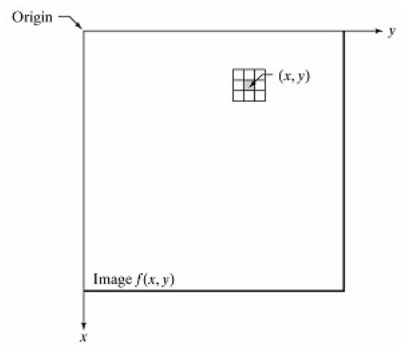
Chapter10: Image Processing in the Spatial/Frequency Domains and Edge Detection

Processing in Spatial-Domain:



Operations in the spatial domain follows a generic formula:

$$g(x,y) = T[f(x,y)] \tag{10.1}$$

If the support region is only a single pixel then they are known as **point operations**. If it is over a region then called **mask operations**.

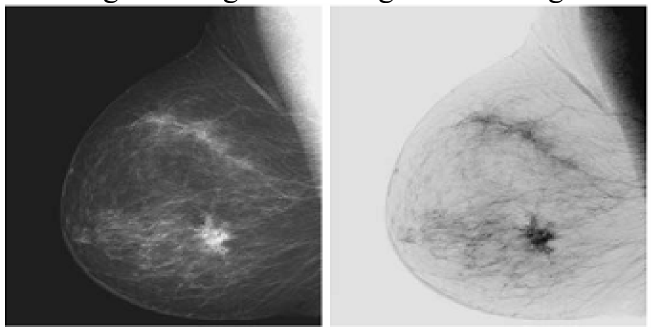
Intensity transformation function:

$$s = T(r) \tag{10.2}$$

where s,r are input and output variables, respectively.

Examples:

Negation: Below we have the original image from a digital mammogram and its negated version.



Pixel differencing: Finding vertical edges: An edge is a junction between two pixels where there is a significant change in gray-level. A simple operation to find vertical edges is just to subtract each pixel's gray-level from that on its right. If changes bigger than, say, 10 gray levels are regarded as significant, then we can produce a binary image indicating where there are edges. Consider a 3x4 segment of image:

	4	5	6	7
4	14	17	20	21
5	19	30	35	36
6	18	45	40	38

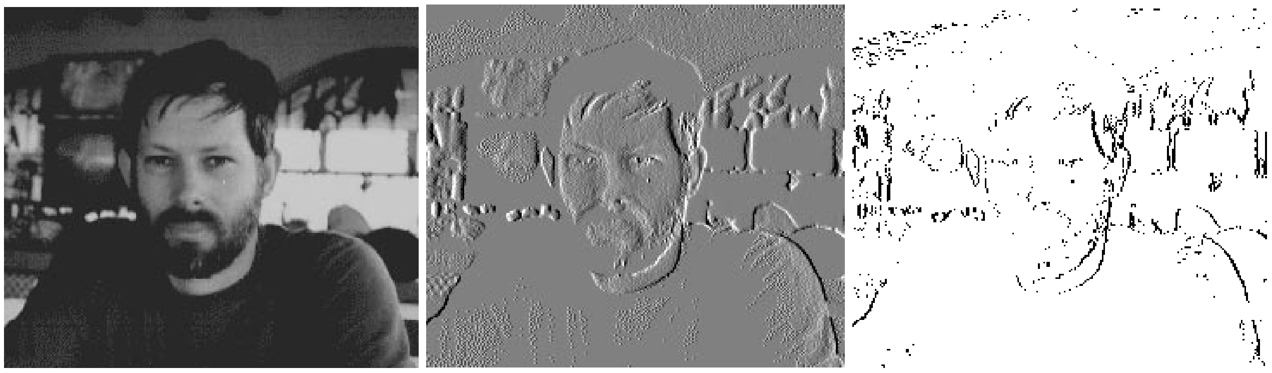
$\rightarrow$

5	6	7
3	3	1
11	5	1
27	-5	-2

$\rightarrow$

5	6	7
0	0	0
1	0	0
1	0	0

Note that we have to make an arbitrary choice to put the result of subtracting (4,4) from (5,4) into (5,4). The second operation is thresholding. Choosing the threshold (in this case 10) is a problem in its own right. If we want to detect all vertical edges, then we need to threshold so that we record a "1" wherever the difference is greater than 10 or less than -10. These two operations, though simple, in fact conform to the structure used by a very wide range of visual systems.

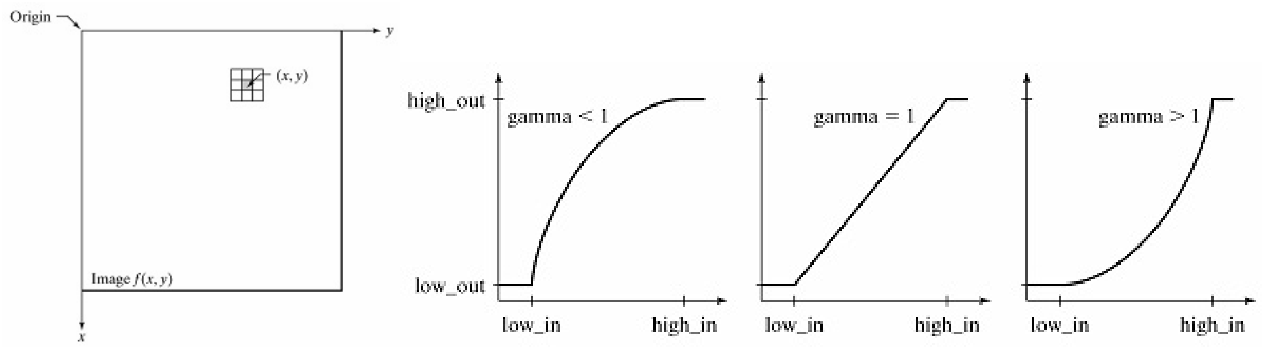

Original 8-bit image Pixel differences One-bit thresholded image (edges)

Log and Power LawTransformations: They are defined by one of the following two expressions:

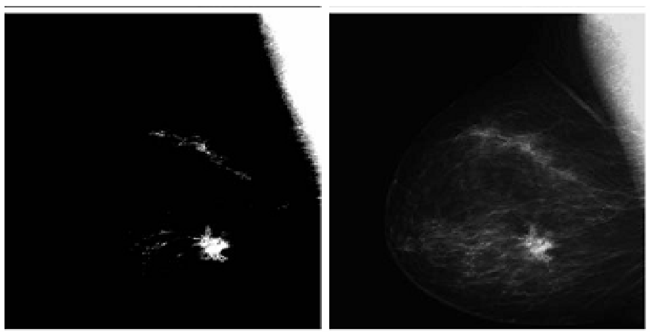
$$s = c.\log(1 + r) \qquad (10.3)$$

$$s = c.r^{\gamma} \qquad (10.4)$$

where r is the input and s is the output of the transformation, r is constant (usually 1.0) and $r \geq 0$.



Below we display two pixel level enhancements: Pixel intensity expansion in the range [0.25,0.75] and the power transformation with the $\gamma = 2.0$ from mammogram pictures above.



3. Histogram Processing: Probability approximations are found for each pixel by computing first the frequency of occurrences of each gray level (histogram): $h(r_k) = n_k / n \quad for\ k = 0,1,\cdots,L-1$ and then they are normalized to obtain probabilities. They could be histogram points, bar graphs, discrete impulses (stem) or continuous plots.

Transformation based on histogram equalization: Given that we know the pdf (or histogram) for the pixel intensity in an image, the histogram equalized output is the cdf (area under the pdf curve (accumulated histogram values):

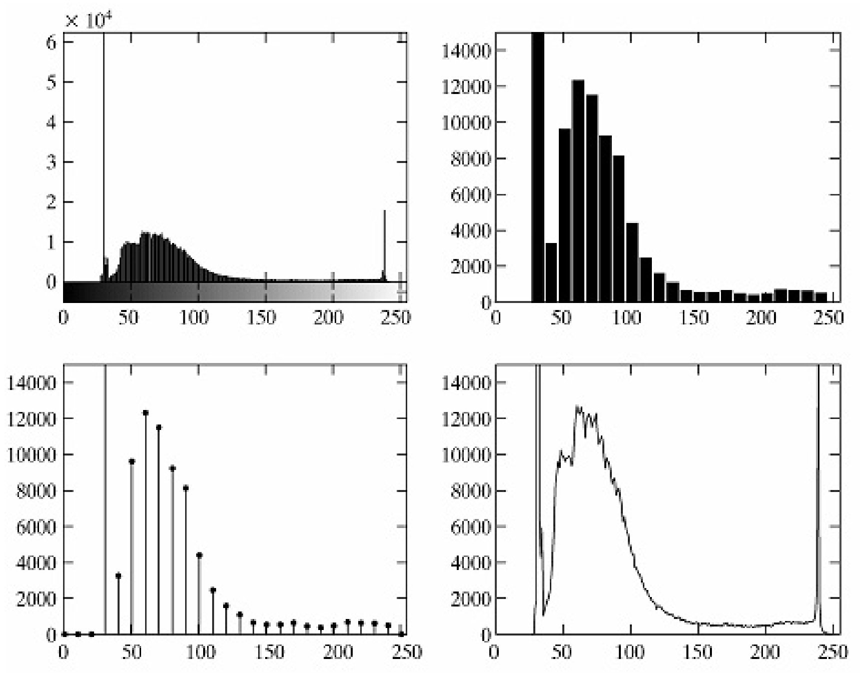
Continuous case: $$s = T(r) = \int_0^r p_r(u).du \qquad (10.5a)$$

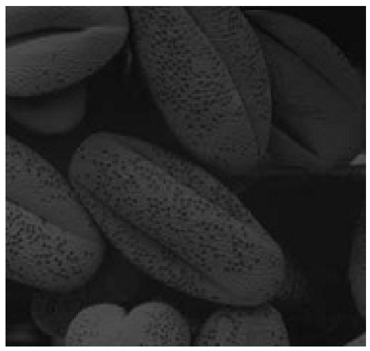
Discrete case: $$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} n_j / n \quad for\ k = 0,1,\cdots,L-1 \qquad (10.5b)$$

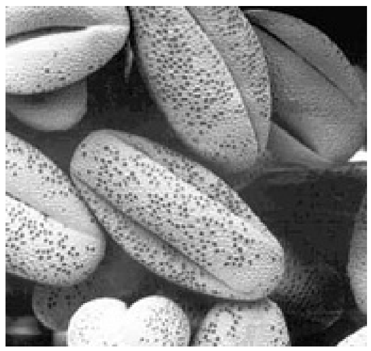
Inverse transformation can be used to come back to the original case, i.e., invertible operation:

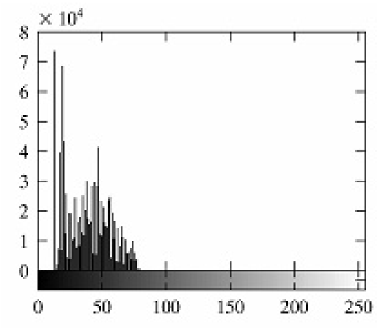
$$r_k = T^{-1}(s_k) \qquad (10.5c)$$

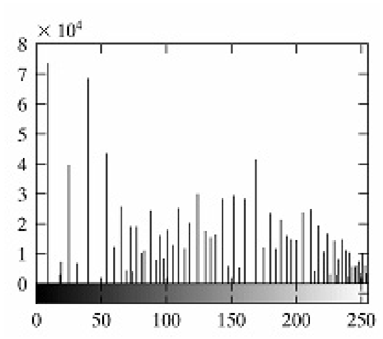
There is a neat 5-point algorithm for implementing histogram equalization including examples in GW p:99-102:







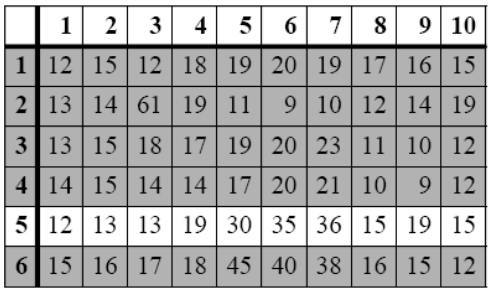




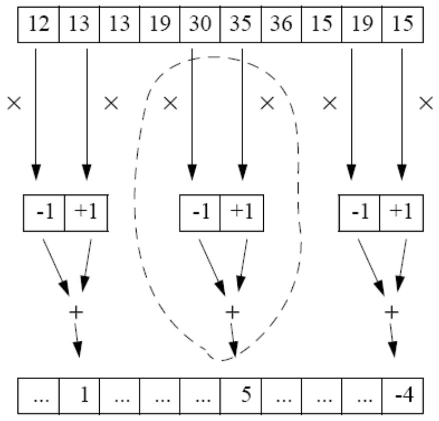
4. Image enhancement with convolution masks with *m* rows and n columns:

$$g(x,y)=\sum_{s=-a}^{a}\sum_{u-b}^{b} w(s,u).f(x+s,y+u) \quad \text{with } a=\frac{m-1}{2};\, b=\frac{n-1}{2} \qquad (10.6)$$

Consider one row of the image and a horizontal mask, above equation is operated only over a single row:

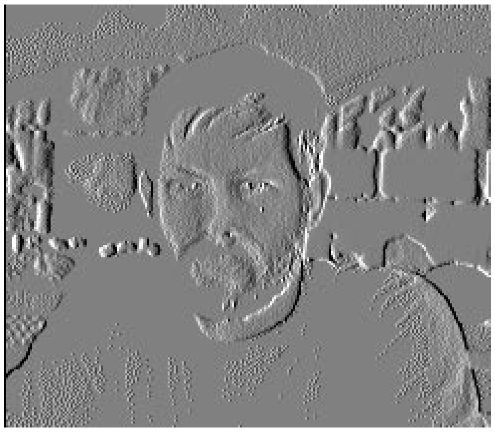


	1	2	3	4	5	6	7	8	9	10
1	12	15	12	18	19	20	19	17	16	15
2	13	14	61	19	11	9	10	12	14	19
3	13	15	18	17	19	20	23	11	10	12
4	14	15	14	14	17	20	21	10	9	12
5	12	13	13	19	30	35	36	15	19	15
6	15	16	17	18	45	40	38	16	15	12



-1	**+1**

The impact on the image is smoothing (averaging) resulting in emphasizing vertical edges.

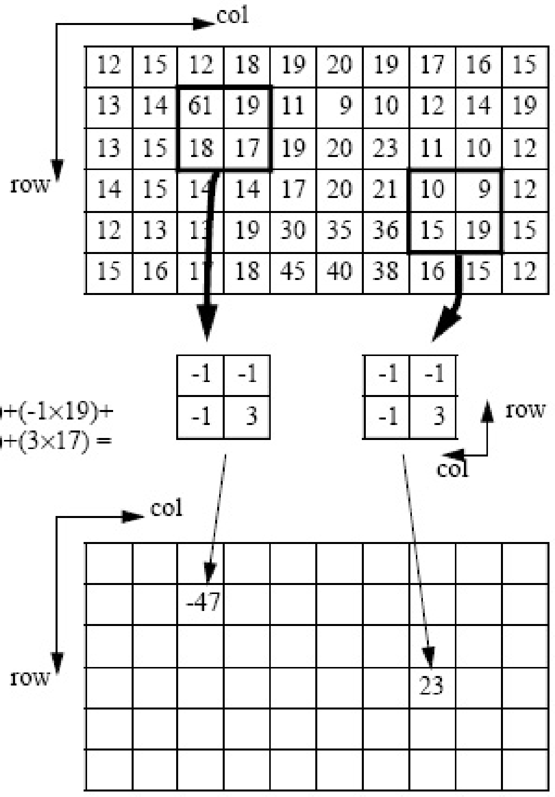


- Masks could be vertical, which results in vertical edge enhancement.

- Masks could be more than 2 columns or rows, combining a larger range of pixel values. For example the example below combines some smoothing and horizontal differencing:

-1	-2	0	+2	+1

- Mask can be 2-dimensional—in fact, just like a small piece of image.



12	15	12	18	19	20	19	17	16	15
13	14	61	19	11	9	10	12	14	19
13	15	18	17	19	20	23	11	10	12
14	15	14	14	17	20	21	10	9	12
12	13	13	19	30	35	36	15	19	15
15	16	17	18	45	40	38	16	15	12

-1	-1
-1	3

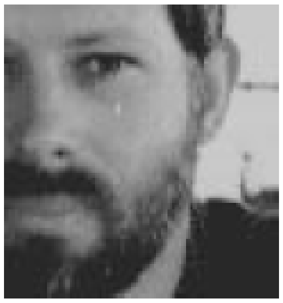
- Diagonal Differences:

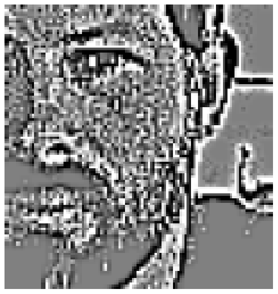
-1	0
0	+1

0	-1
+1	0

- Center surround mask (averaging, smoothing);

-1/8	-1/8	-1/8
-1/8	+1	-1/8
-1/8	-1/8	-1/8

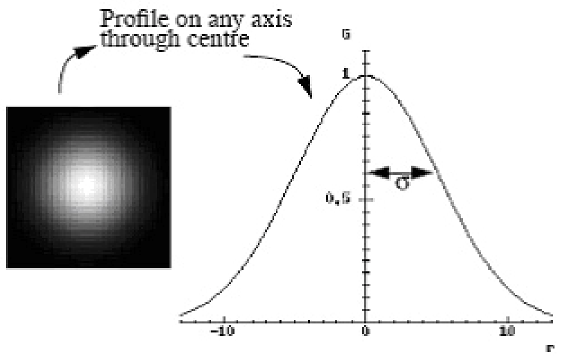


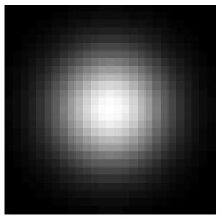


- The example below also combines smoothing and differencing, and is known as the **Sobel operator**.

-1	0	+1
-2	0	+2
-1	0	+1

Gaussian smoothing mask: A more effective smoothing mask falls off gently at the edges, where its width is described by the standard deviation σ.



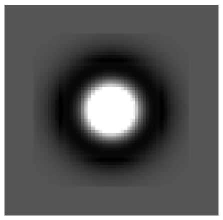


$*$

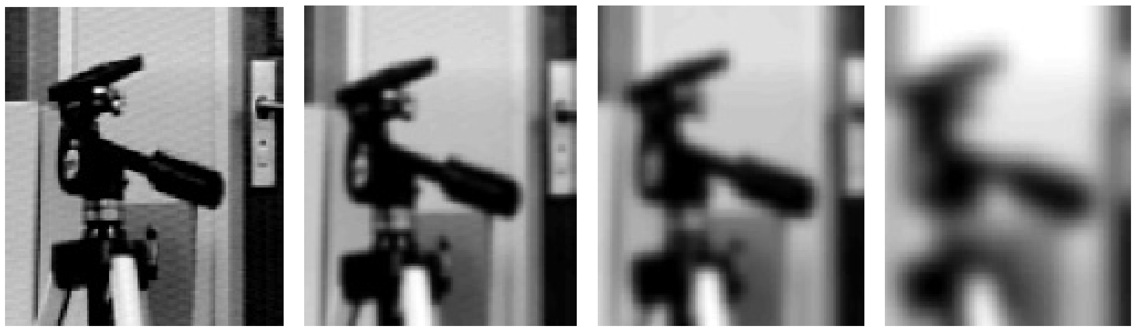
convolve

-1/8	-1/8	-1/8
-1/8	+1	-1/8
-1/8	-1/8	-1/8

$\longrightarrow$



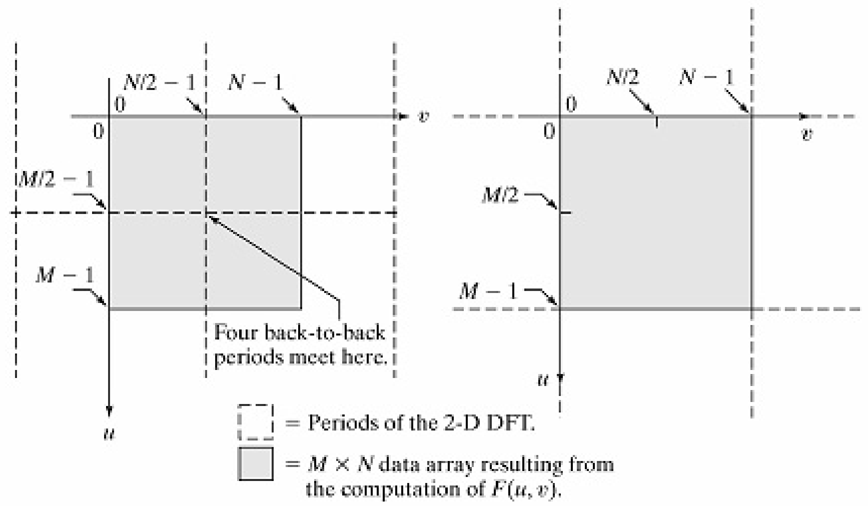
They are effectively used for removing any texture with a scale smaller than the mask dimensions. Small-scale texture is said to have a high *spatial frequency*. Smoothing removes this, leaving low spatial frequencies. Here is the effect of progressively increasing s. Its values are 1, 2 and 4 in the 3 smoothed images.



- In addition, there are median filtering masks and Laplacian masks used in spatial-domain image enhancement.

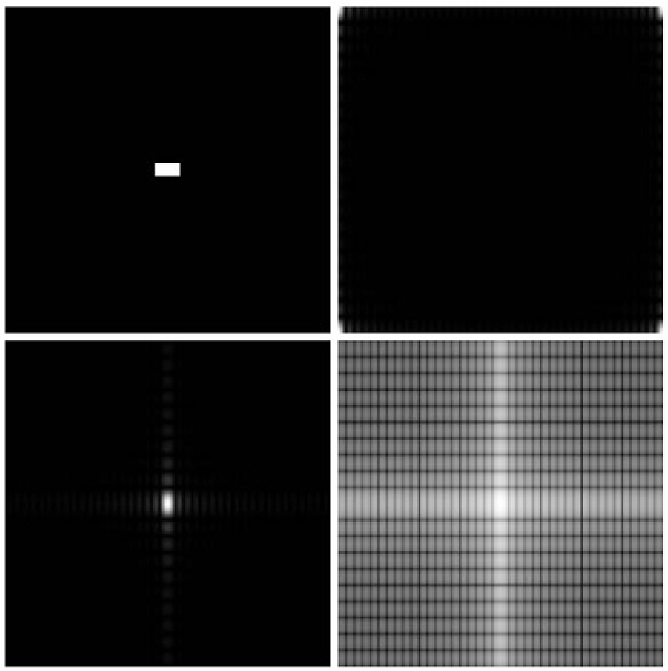
Processing in Frequency-Domain: FT for an MxN image is given by:

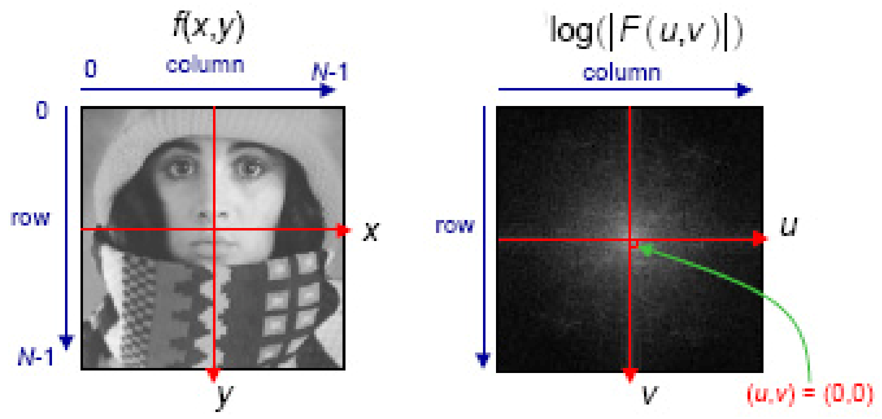
$$F(u,v) = \frac{1}{M.N} \cdot \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \tag{10.7}$$


N/2 − 1
N − 1
0
v
M/2 − 1
M − 1
Four back-to-back periods meet here.
u
N/2
M/2
= Periods of the 2-D DFT.
= M × N data array resulting from the computation of F(u, v).

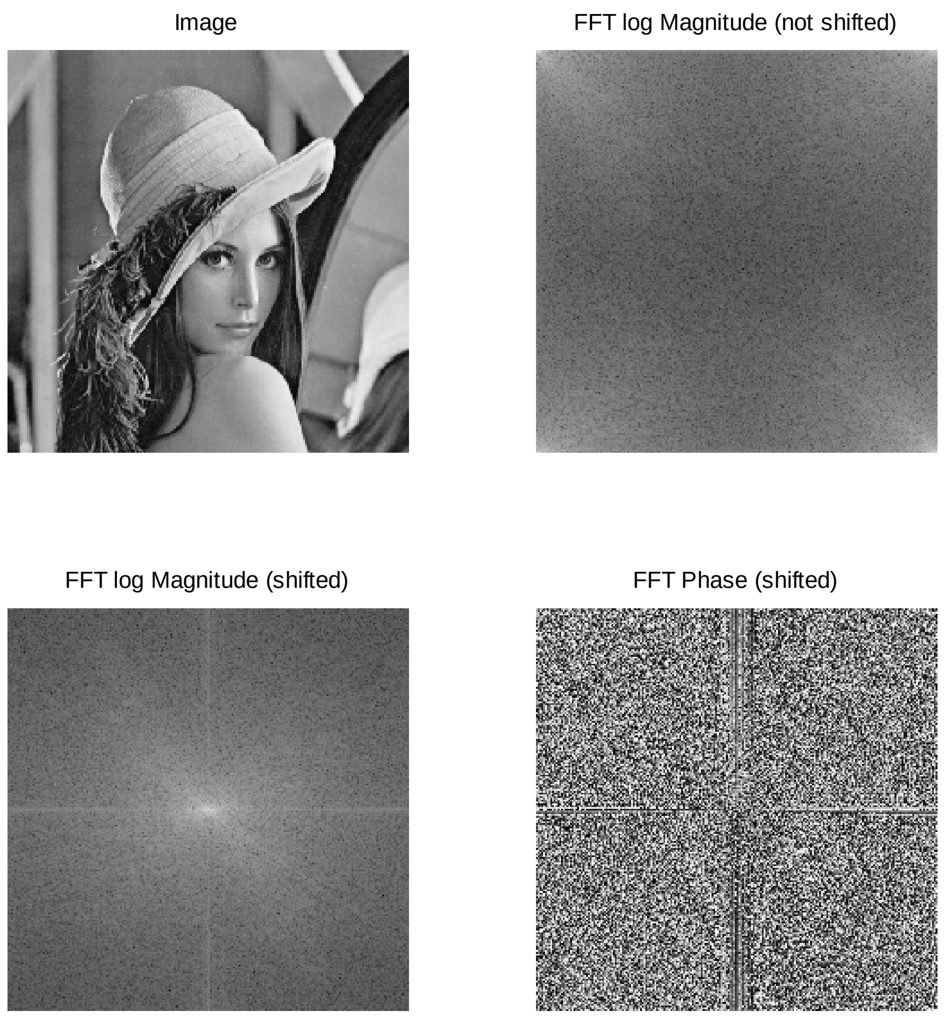
a b

FIGURE 4.2 (a) $M \times N$ Fourier spectrum (shaded), showing four back-to-back quarter periods contained in the spectrum data. (b) Spectrum obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ prior to computing the Fourier transform. Only one period is shown shaded because this is the data that would be obtained by an implementation of the equation for $F(u, v)$.





Lena (512x512) 8-bits/pixel original. FFT log magnitude without placing (0,0) in the center, shifted version (commonly known as the DFT) and the corresponding phase spectrum. (using 2-D fft algorithm in Image Processing toolbox in Matlab.)



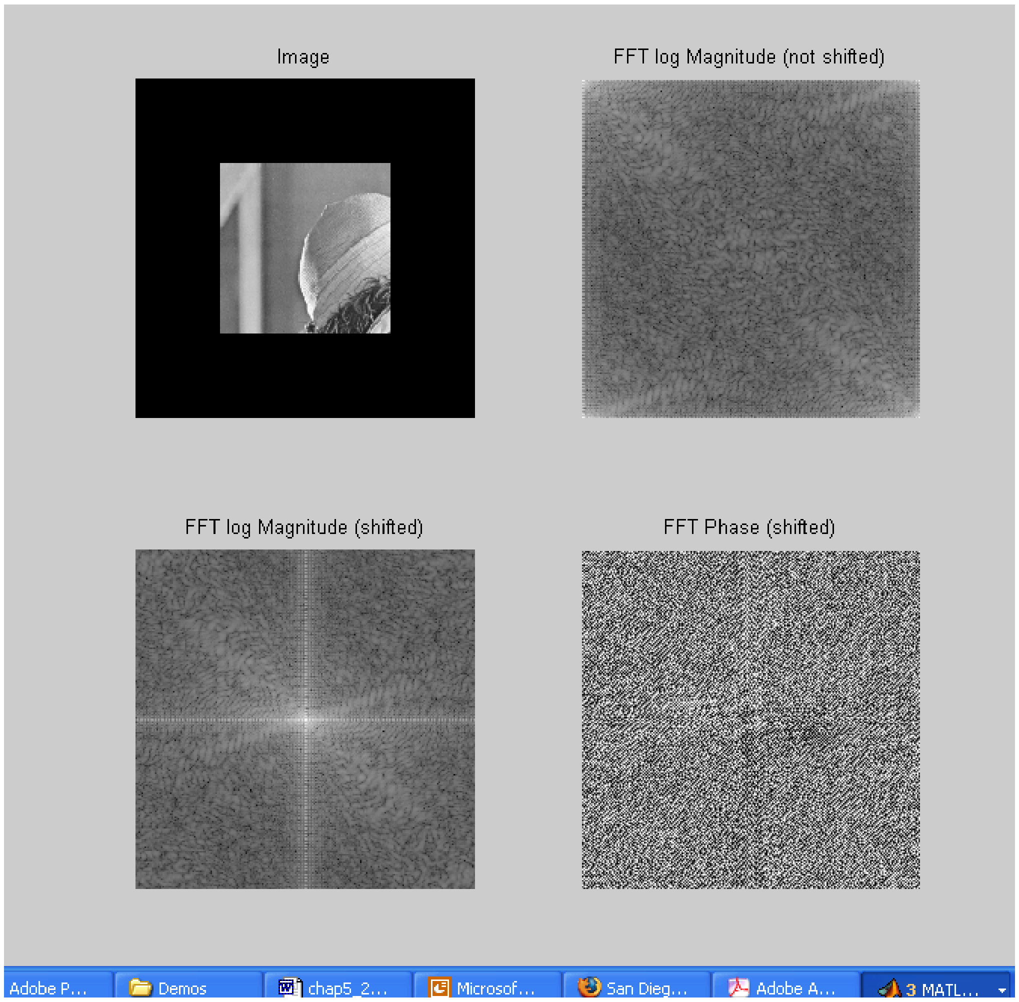
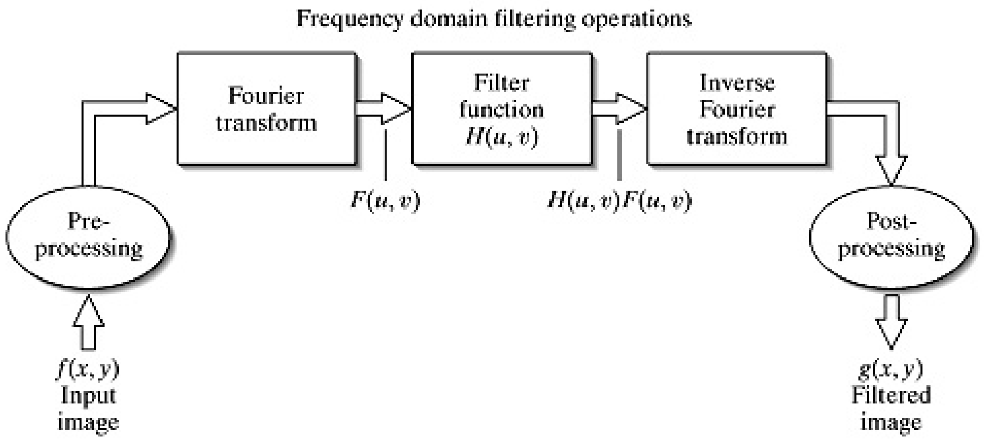
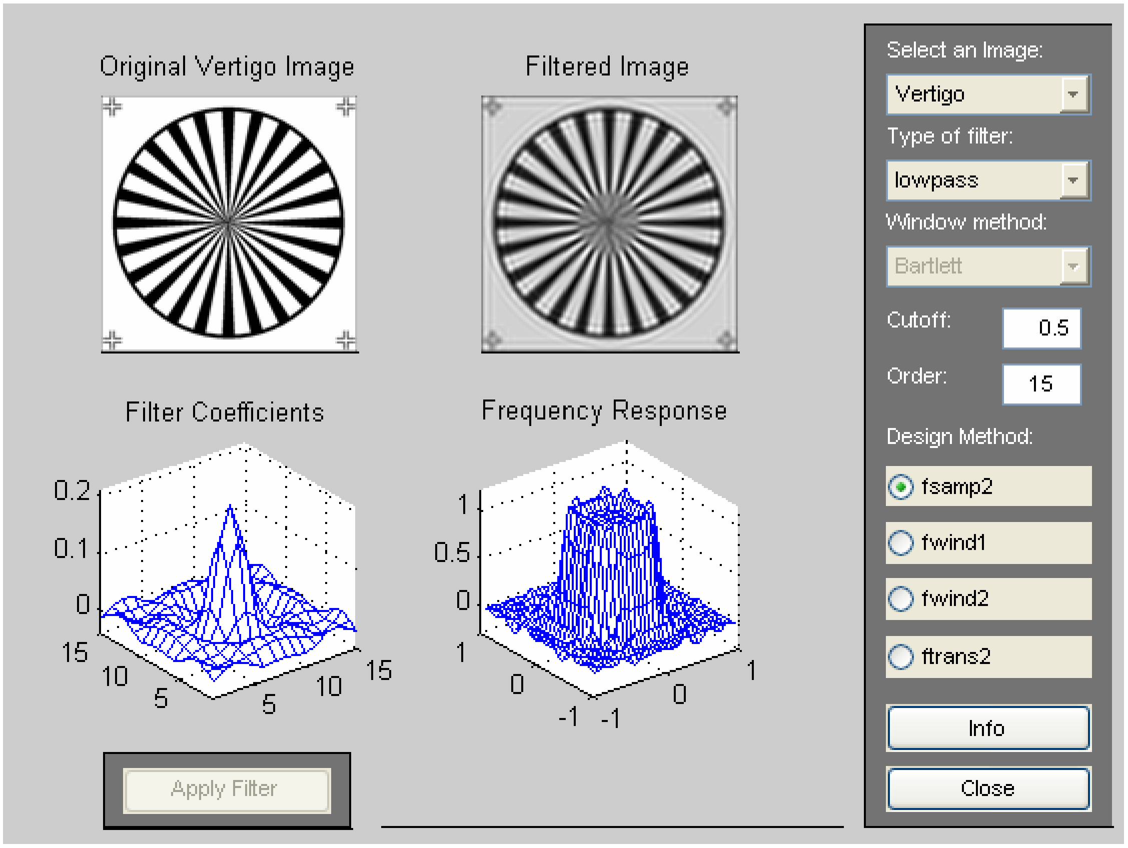


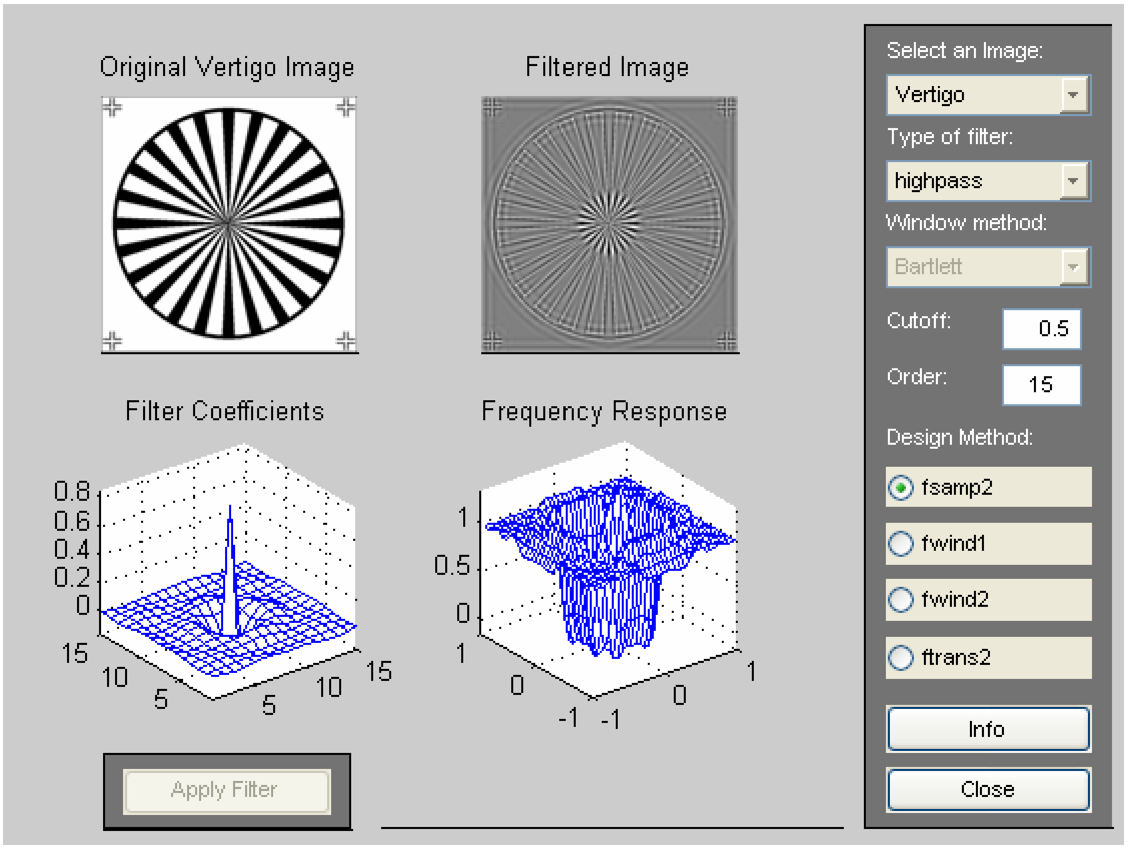
Image Enhancement via Filtering



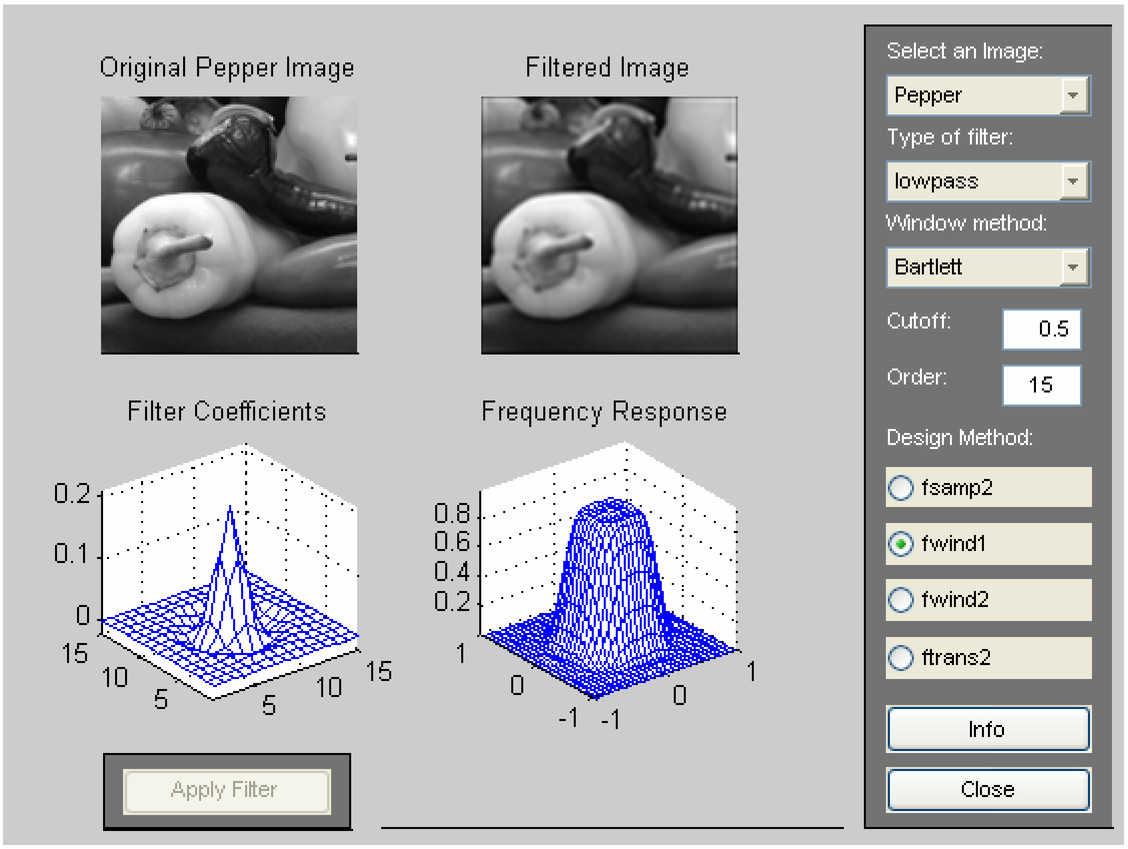
Even more critical than the case in 1-D signal filtering, fast convolution is ubiquitously used to implement filters in image processing applications. Recall that 2 forward FFTs, 2 inverse FFTs and a point-by-point image multiply in the frequency-domain will be at a significantly smaller fraction of the time required to perform a 2-D comnvolution.



Select an Image: Vertigo
Type of filter: lowpass
Window method: Bartlett
Cutoff: 0.5
Order: 15
Design Method: [✓ fsamp2] fwind1 fwind2 ftrans2



Select an Image: Vertigo
Type of filter: highpass
Window method: Bartlett
Cutoff: 0.5
Order: 15
Design Method: [✓ fsamp2] fwind1 fwind2 ftrans2



Select an Image: Pepper

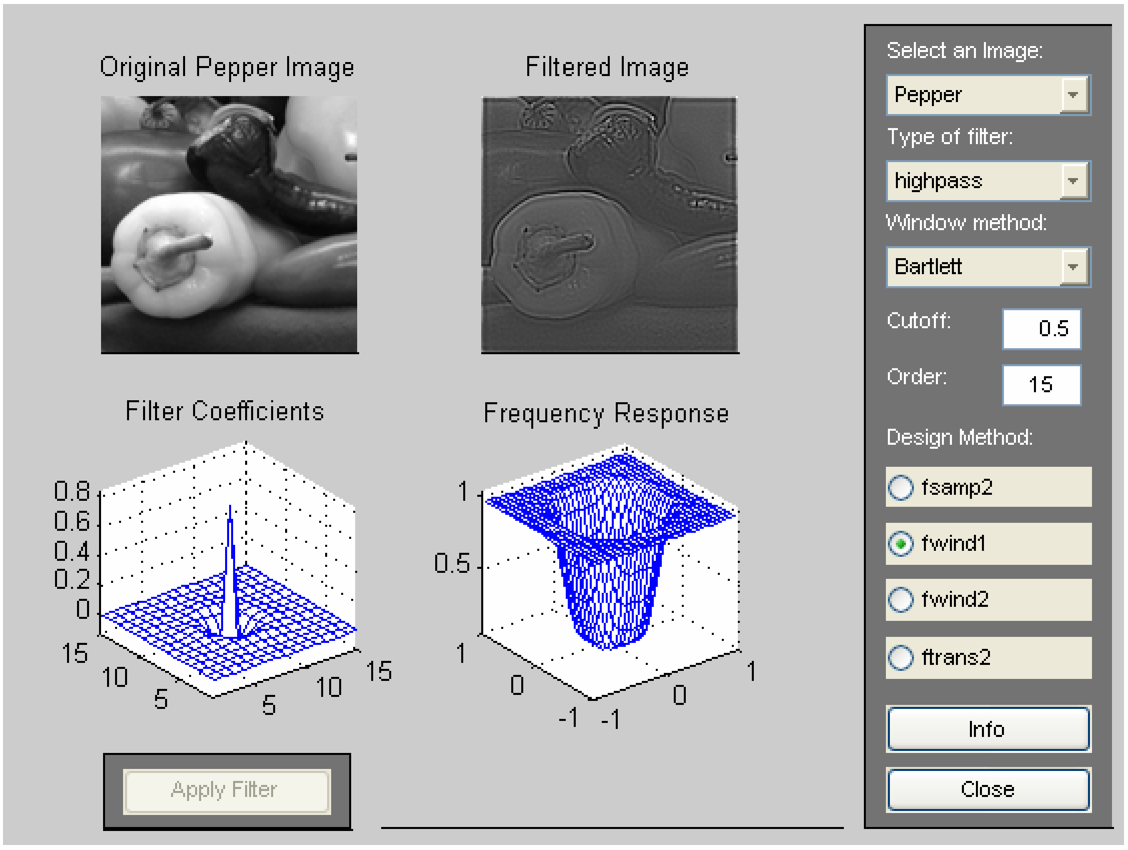
Type of filter: lowpass

Window method: Bartlett

Cutoff: 0.5

Order: 15

Design Method: ( ) fsamp2 [✓ fwind1] ( ) fwind2 ( ) ftrans2



Select an Image: Pepper

Type of filter: highpass

Window method: Bartlett

Cutoff: 0.5

Order: 15

Design Method: ( ) fsamp2 [✓ fwind1] ( ) fwind2 ( ) ftrans2

Ideal low-pass using rectangular spatial window:

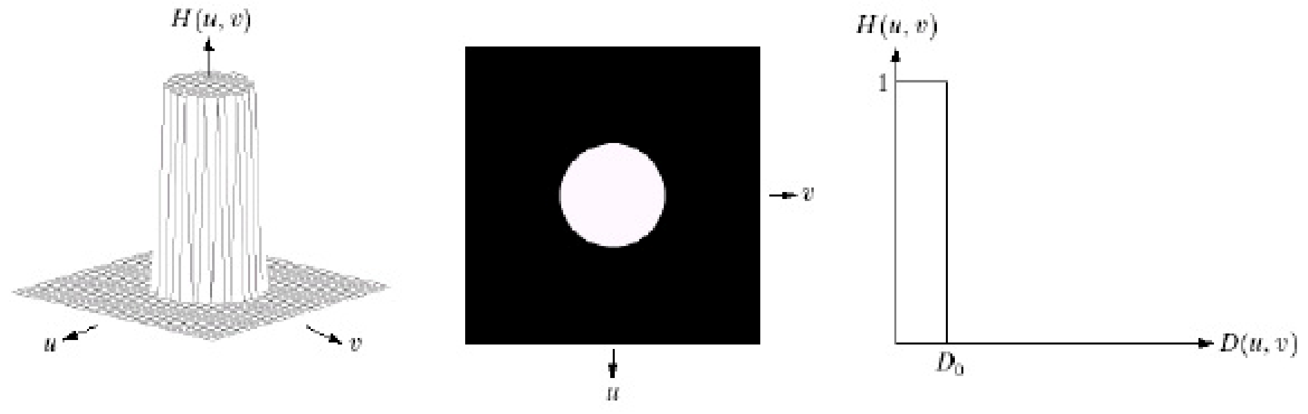
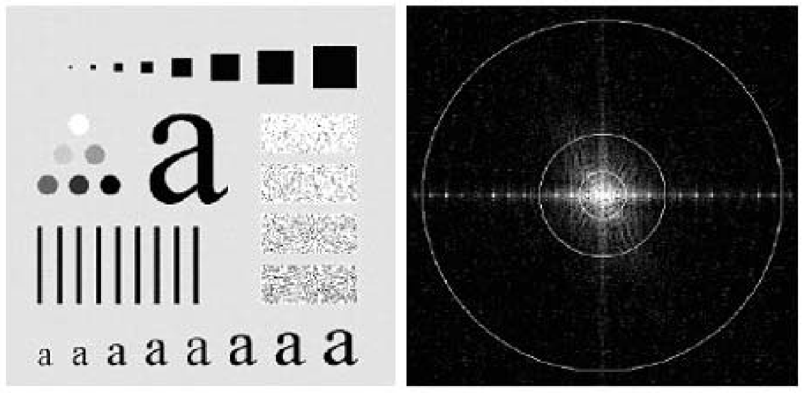


FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Power Distribution through 2-D DFT in images:



a b

FIGURE 4.11 (a) An image of size 500 × 500 pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.

Ideal low-pass filtering results in blurring due to smoothing (averaging) and ringing due to Gibbs phenomenon of representing using a finite number of terms in representing an ideal rectangular pulse.

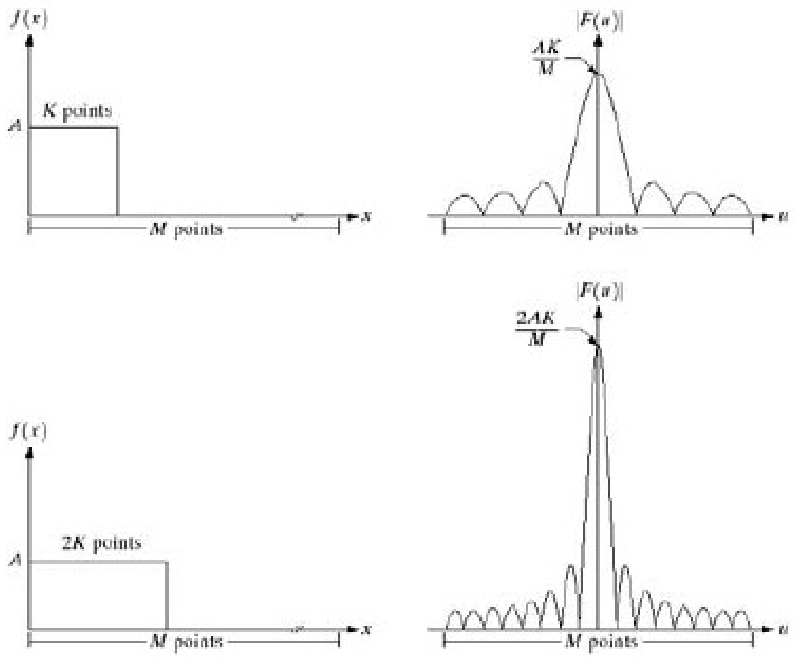


FIGURE 4.2 (a) A discrete function of M points, and (b) its Fourier spectrum. (c) A discrete function with twice the number of nonzero points, and (d) its Fourier spectrum.

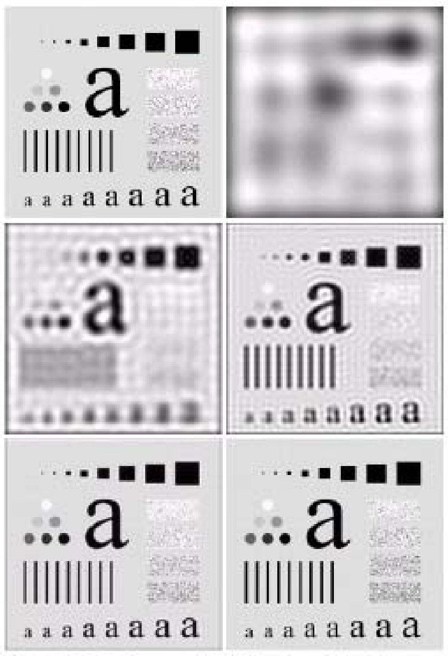
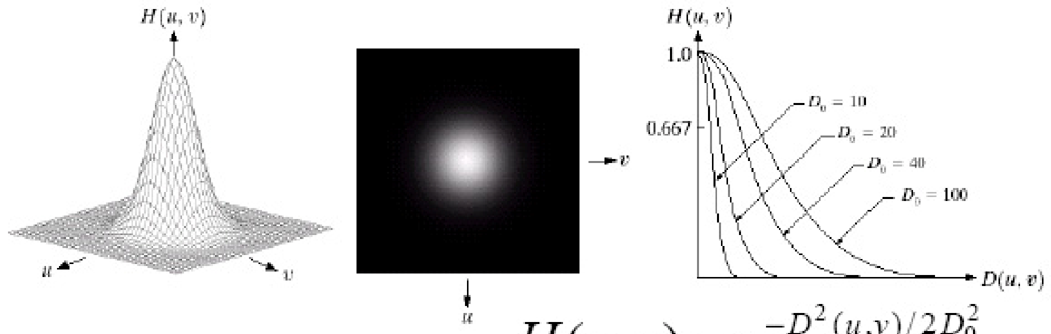


FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

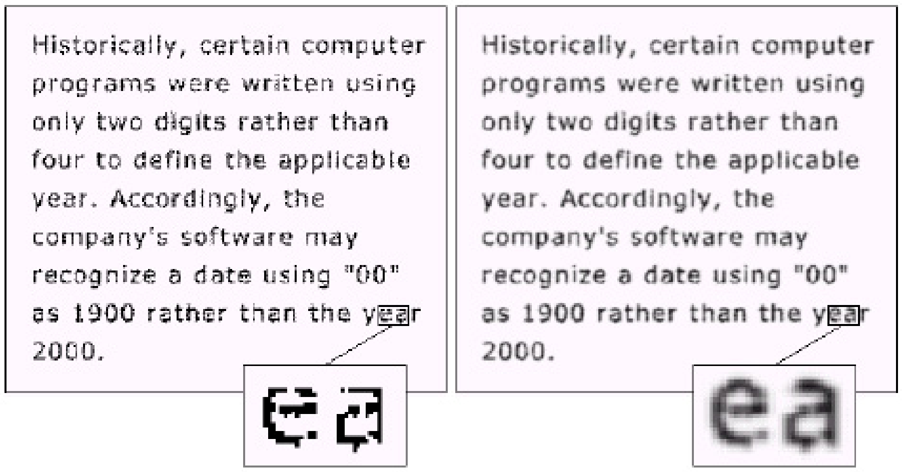
Low-pass filtering using Gaussian Low-pass filter characteristics.



$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

FIGURE 4.17 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

Application to Text Smoothing:



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

FIGURE 4.19 (a) Sample text of poor resolution (note broken characters in magnified view). (b) Result of filtering with a GLPF (broken character segments were joined).

Low-pass filtering using Butterworth Low-pass filter characteristics:

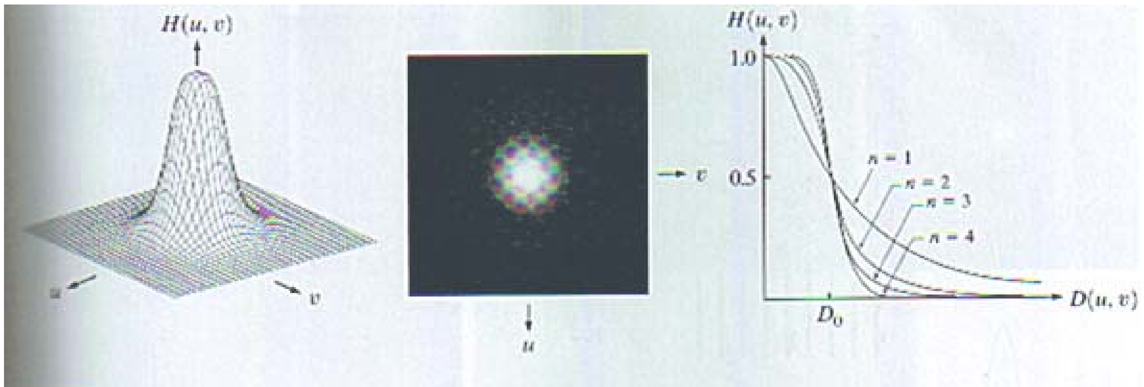


FIGURE 4.14 (a) Perspective plot of a Butterworth lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.

High-pass filtering using ideal, Butterworth and Bartlett windows:

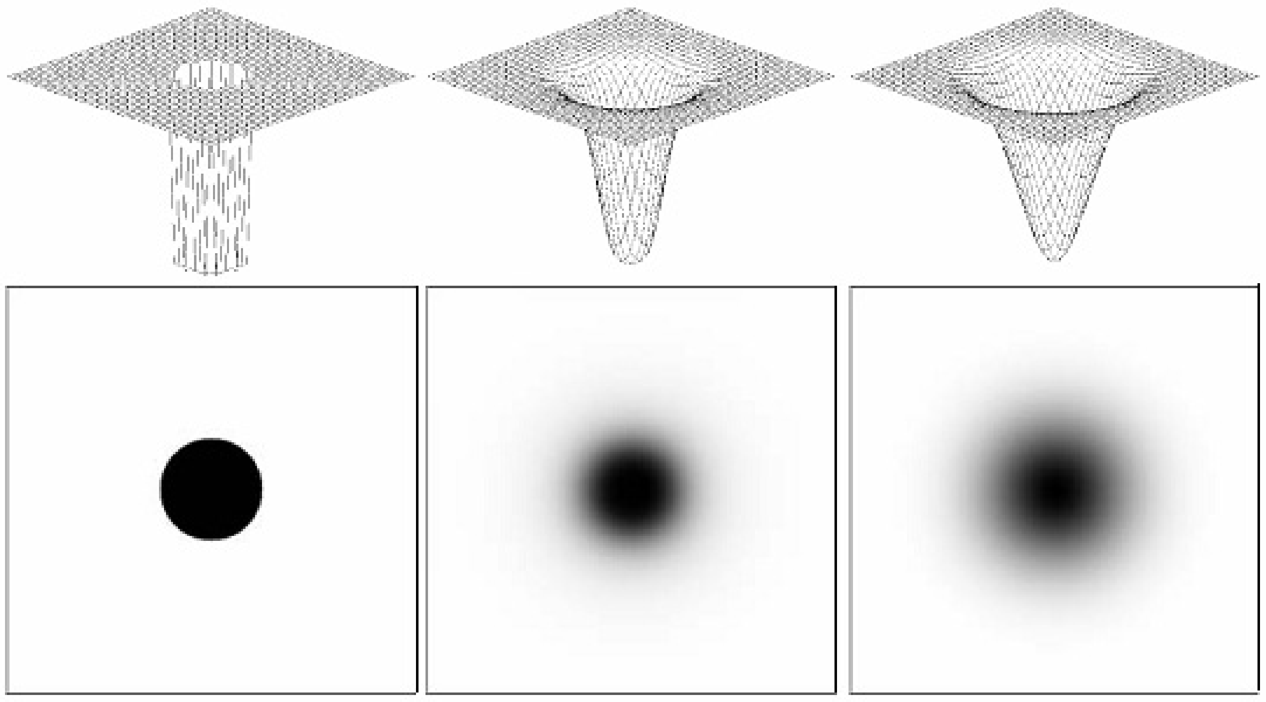


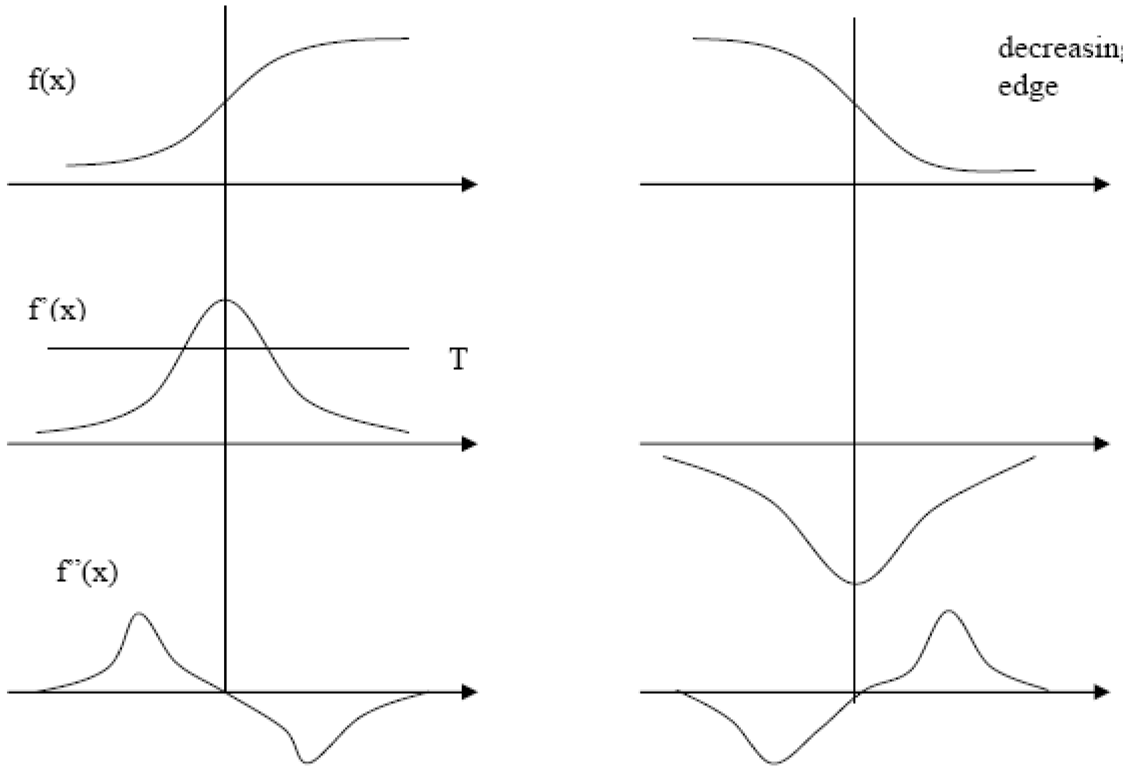
FIGURE 4.17 Top row: Perspective plots of ideal, Butterworth, and Gaussian highpass filters. Bottom row: Corresponding images.

Padding Images with zeros is very frequently used for computational efficiency gains. For instance an image of size 480x640 will be processed significantly faster via fast convolution using 2-D FFT by padding both rows and columns with zeros to bring to a 512x1024, i.e., (2^9x2^{10}) and then doing 2-D IFFT. Otherwise, we would need 480x640 double convolution or IDFT, which are both equally costly by a couple of orders of power!

Edge Detection

General Principles:

In a continuous 1-D signal *f(x),* an edge is defined as the point where the derivative of the signal *f'(x)* has an extremum, or equivalently where the second derivative *f"(x)* is zero, as shown below.



Task: Extend these concepts to:

i) two-dimensions, and
ii) discrete images.

The first can be accomplished by replacing the first and second derivatives with the gradient and Laplacian operators, respectively.

Approximating the first and second derivatives (partials) with the appropriate finite difference operators can do the latter.

Because the derivative (hence finite difference) operators are indeed high-pass filters, edge detection is very sensitive to noise. Two types of errors result due to noise:

1. **False positives:** Noise may generate many small peaks in the magnitude of the gradient resulting in false edges.
2. **False negatives:** Noise may result in shifts in true edge locations, resulting in missing actual edge pixels.

There are many different edge detection algorithms, which distinguish themselves on how they deal with these problems.

Gradient-Based Methods:

Gradient vector for a continuous image $f_c(x_1, x_2)$ is defined as:

$$\nabla f_c(x_1,x_2) = \begin{bmatrix} \dfrac{\partial f_c(x_1,x_2)}{\partial x_1} \\ \dfrac{\partial f_c(x_1,x_2)}{\partial x_2} \end{bmatrix} \tag{10.8a}$$

and its magnitude: $|\nabla f_c(x_1,x_2)| = \sqrt{[\frac{\partial f_c(x_1,x_2)}{\partial x_1}]^2 + [\frac{\partial f_c(x_1,x_2)}{\partial x_2}]^2}$ (10.8b)

Edge detection using gradient-based techniques normally employ a thresholding mechanism:

$$|\nabla f_c(x_1,x_2)| \geq T \tag{10.9}$$

where T is a threshold, which is determined in many applications from the image database at hand.

Discrete Approximations:

Forward Difference: $\frac{\partial f_c(x_1,x_2)}{\partial x_1} \approx f(n_1+1,n_2) - f(n_1,n_2)$ (10.10a)

Backward Difference: $\frac{\partial f_c(x_1,x_2)}{\partial x_1} \approx f(n_1,n_2) - f(n_1-1,n_2)$ (10.10b)

Central Difference: $\frac{\partial f_c(x_1,x_2)}{\partial x_1} \approx \frac{1}{2}\{f(n_1+1,n_2) - f(n_1-1,n_2)$ (10.10c)

Average Central Difference:

$$\begin{aligned}\frac{\partial f_c(x_1,x_2)}{\partial x_1} \approx \frac{1}{6}\{&[f(n_1+1,n_2) - f(n_1-1,n_2)] \\ &+[f(n_1+1,n_2-1) - f(n_1-1,n_2-1)] \\ &+[f(n_1+1,n_2+1) - f(n_1-1,n_2+1)]\}\end{aligned} \tag{10.10d}$$

These computations can be interpreted as passing FIR filters, that is, 2-D convolution of an image with an impulse response. If $h_1(n_1,n_2)$ and $h_2(n_1,n_2)$ denote the filter impulse responses that approximate the horizontal and vertical partials, respectively. Then, the gradient of a discrete image can be written as:

$$\nabla f_c(n_1,n_2) = \begin{bmatrix} f_{x_1}(n_1,n_2) \\ f_{x_2}(n_1,n_2) \end{bmatrix} = \begin{bmatrix} f(n_1,n_2) ** h_1(n_1,n_2) \\ f(n_1,n_2) ** h_2(n_1,n_2) \end{bmatrix} \tag{10.11}$$

where $**$ represents a 2-D convolution, i.e., convolution over columns followed by another convolution over rows. The magnitude and the direction of the gradient vector at point n_1, n_2 are given by:

$$|\nabla f_c(n_1,n_2)| = \sqrt{[f_{x_1}(n_1,n_2)]^2 + [f_{x_2}(n_1,n_2)]^2} \tag{10.12a}$$

$$\angle \nabla f(n_1,n_2) = \arctan\left(\frac{f_{x_2}(n_1,n_2)}{f_{x_1}(n_1,n_2)}\right) \tag{10.12b}$$

With the following mask, we discuss some of the commonly used FIR filter kernels for $h_1(n_1, n_2)$ and $h_2(n_1, n_2)$. The **Prewitt kernel** represents the average central difference approximation. The kernels for the partials in x_1 and x_2 directions are given by:

-1	0	1
-1	0	1
-1	0	1

1	1	1
0	0	0
-1	-1	-1

These look like **Sobel kernel** (mask), where the latter applies twice the weight to the center row horizontally and vertically given by:

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Isotropic filters must not favor any particular edge direction. Prewitt and Sobel filters respond to diagonal edges differently than the horizontal and vertical edges because their coefficients do not take into account larger pixel distances in the diagonal directions. The Prewitt filter is less sensitive to diagonal edges than to horizontal and vertical ones, while the opposite is true for the Sobel filter.

Roberts Kernel:

0	1
-1	0

1	0
0	-1

Laplacian-Based Methods:

Laplacian of a continuous image is defined as the dot product of the gradient by itself:

$$\nabla^2 f_c(x_1, x_2) = \nabla \circ \nabla f_c(x_1, x_2) = \frac{\partial^2 f_c(x_1, x_2)}{\partial x_1^2} + \frac{\partial^2 f_c(x_1, x_2)}{\partial x_2^2} \qquad (10.13)$$

The Laplacian is isotropic favoring no particular edge direction. In order to compute a discrete approximation to Laplacian, we can use the forward difference to approximate the first derivative,

$$\frac{\partial f_c(x_1, x_2)}{\partial x_1} \approx f(n_1+1, n_2) - f(n_1, n_2) \qquad (10.14)$$

and then the backward difference to approximate the second derivative as the derivative of the first derivative:

$$\begin{aligned} \frac{\partial}{\partial x_1}\left[\frac{\partial f_c(x_1, x_2)}{\partial x_1}\right] &\approx [f(n_1+1, n_2) - f(n_1, n_2)] - [f(n_1, n_2) - f(n_1-1, n_2)] \\ &= f(n_1+1, n_2) - 2.f(n_1, n_2) + f(n_1-1, n_2) \end{aligned} \qquad (10.15)$$

When we place this and the corresponding one for the other variable x_2 to get an approximation to the Laplacian equation defined above:

$$\nabla^2 f_c(x_1, x_2) \approx f(n_1+1, n_2) + f(n_1-1, n_2) + f(n_1, n_2+1) + f(n_1, n_2-1) - 4.f(n_1, n_2) \qquad (10.16)$$

This last equation for the Laplacian process is normally expressed as an FIR filter with an impulse response:

0	1	0
1	-4	1
0	1	0

However, depending upon the choice of derivative approximations other FIR filter kernels can also be obtained, such as:

1	1	1
1	-8	1
1	1	1

-1	2	-1
-2	-4	2
-1	2	-1

Laplacian of Gaussian (LoG) Filter: It is a Gaussian filter with the scale parameter σ:

$$h_c(x_1, x_2) = \frac{x_1^2 + x_2^2 - 2\sigma^2}{\sigma^4}.\exp\{-\frac{x_1^2 + x_2^2}{2\sigma^2}\} \tag{10.17}$$

Digital implementation of the **LoG** filter requires sampling $h_c(x_1, x_2)$ on a large enough support for a particular value of σ. It is important to note that the LoG filter is separable, hence can be efficiently implemented as a cascade of two 1-D filters.

Canny Edge Detection: Canny's edge detection procedure is among the most widely used employing elements of both gradient-based edge detection and Gaussian filtered scale space. It consists of the following steps:

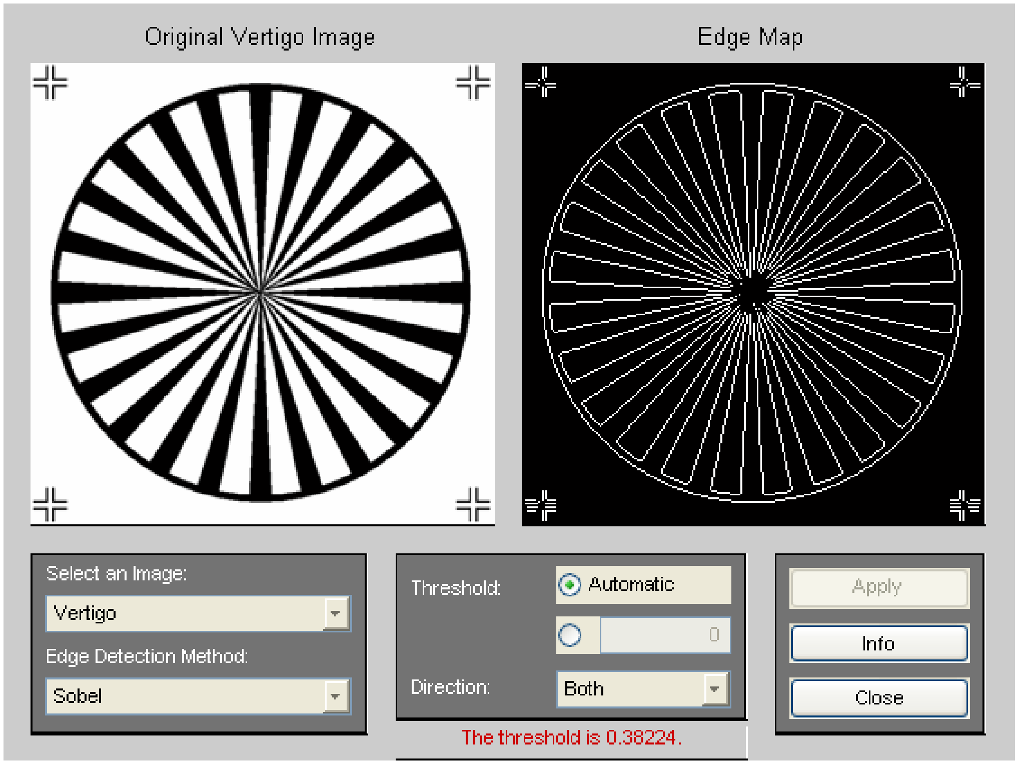
1. Image smoothing with a Gaussian filter with a scale parameter σ.
$$h(n_1, n_2) = K.\exp\{-\frac{n_1^2 + n_2^2}{2\sigma^2}\} \tag{10.18}$$
2. Find the image gradient, including magnitude and direction, at each pixel, using one of the operators, e.g., Roberts or Sobel.
3. Edge thinning by ***non-maximum suppression*:** Gradient direction is used to thin edges by suppressing any pixel response that is not higher than those of two neighboring pixels on either side of it along the direction of the gradient. The two 8-neighbors of a pixel that are to be compared are found by rounding off the computed gradient direction to one of 0, 45, 90 or 135 degrees.
4. Thresholding by gradient magnitude with ***hysterisis***. Two thresholds, an upper and a lower threshold, are defined for edge detection and edge following, respectively, where the upper threshold is two or three times the lower threshold. The edge detection is based on the upper threshold. However, once started a contour segment may be followed through pixels whose gradient magnitude exceeds the lower threshold.
5. A set of edge maps over a range of scales can be produced by varying σ. Fine-to-coarse feature synthesis is used to combine edges at different scales into a single edge map.

References

1. Handbook of Image and Video Processing, ed. Al Bovik, Academic Press, 2000.
2. L. Shapiro and G. Stockman, Computer Vision, Prentice Hall, 2001.
3. J. Canny, "A computational approach to edge detection," IEEE Trans. on Patt. Anal. Mach. Intel., vol. 8, no. 6, pp. 679-698, 1986.

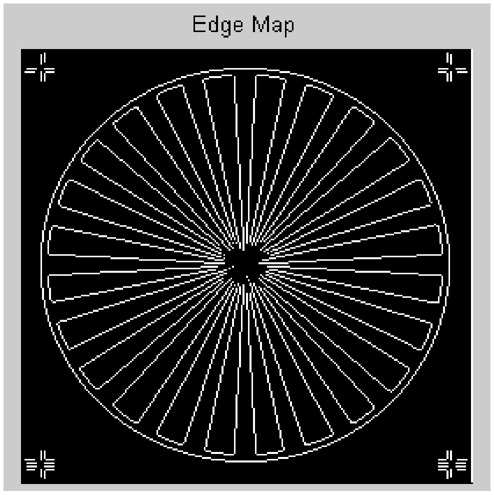
Edge Detection Examples using demos in the Matlab Image Processing Toolbox.

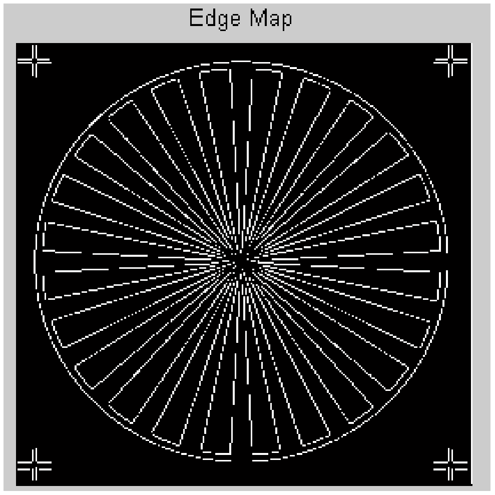
Edge detection on Vertigo image using Sobel kernel with a threshold: 0.38224



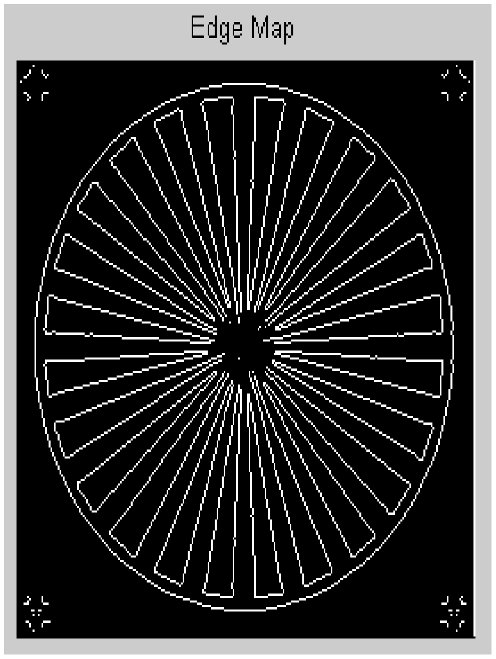
Select an Image: Vertigo
Edge Detection Method: Sobel
Threshold: [✓ Automatic] 0
Direction: Both
The threshold is 0.38224.

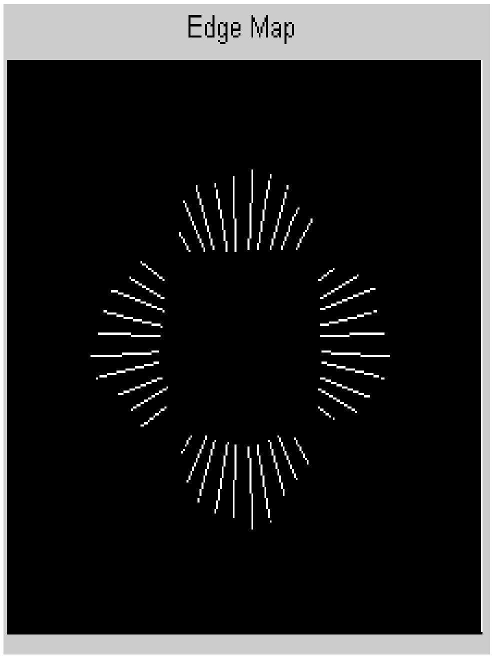
Edge detection on Vertigo image using Prewitt kernel with a threshold: 0.37413 and Roberts kernel with a threshold: 0.57818:



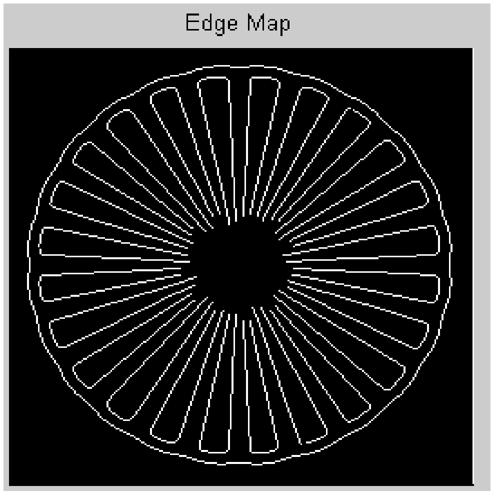


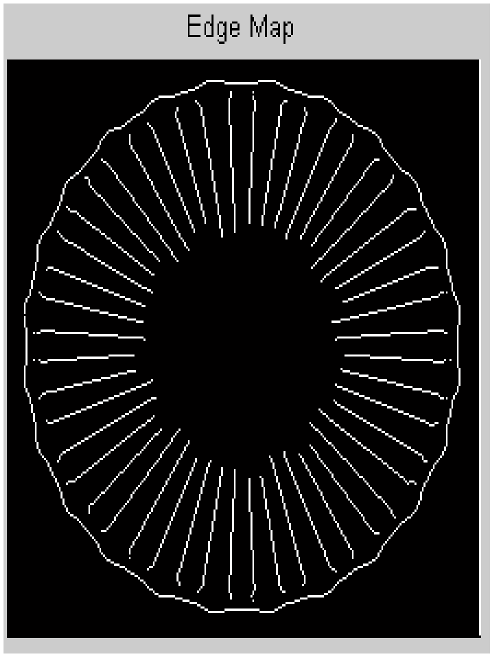
Edge detection on Vertigo image using LoG with sigma: 2 and a threshold: 0.02509, also with sigma: 4 and a threshold: 0.42188





Edge detection on Vertigo image using Canny with sigma: 2 and a threshold: 0.60938 and also with sigma: 4 and a threshold: 0.53125





Similar procedures will be tested with other images. Also Identifying Round Objects will be an integrated application of edge detection, Measuring Angle of Intersection.